4. Message authentication and integrity

Cilj vježbe je demonstrirati korištenje MAC mehanizma za zaštitu integriteta poruke.

1. **zadatak**- štitimo autentičnost sadržaja file pomoću MAC funkcije. U prvom dijelu primamo i šaljemo poruku, a u drugom dijelu autenticiramo.

   Prvi dio- potpisivanje file-a:

   1) otvoriti file i pročitati sadržaj
   2) MAC funkcijom iz sadržaja poruke dobivamo potpis
   3) dodati potpis na poruku

   Drugi dio- verifikacija:

   1) pročitamo sadržaj filea
   2) pročitamo sadržaj filea u kojem je potpis
   3) potpisujemo sadržaj
   4) uspoređujemo novi potpis sa potpisom iz pročitanog file (2)

2. **zadatak**- iz 10 postojećih fileova i njihovih potpisa odrediti koje su transakcije autentične.

```
from pathlib import Path
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.exceptions import InvalidSignature
import re
import datetime

def verify_MAC(key, signature, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
```

```
    try:
        h.verify(signature)
    except InvalidSignature:
        return False
    else:
        return True

#def generate_MAC(key, message):
#   if not isinstance(message, bytes):
#        message = message.encode()

#    h = hmac.HMAC(key, hashes.SHA256())
#    h.update(message)
#    signature = h.finalize()
#    return signature

if __name__ == "__main__":
    # 1. Sign the message
    # 1.1. Read the file content
    # Reading from a file
    #with open("message.txt", "rb") as file:
    #  message = file.read()


    # 1.2. Generate aigning key or secret
    #key = "my super secret".encode()

    # 1.3. Actually sign the message
    #signature = generate_MAC(key=key, message=message)

    # 1.4. Save the signature or MAC tag into a file
    #with open("message.sig", "wb") as file:
    #   file.write(signature)

    # 2. Verify message authenticity
    # 2.1. Read the message file content and the signature
    #with open("message.txt", "rb") as file:
    #    message = file.read()

    #with open("message.sig", "rb") as file:
    #    signature = file.read()

    ## 2.2. Learn/get the signing key
    #key = "my super secret".encode()

    # 2.3. Sign the message and compare locally generates MAC with the received one
    #is_authentic = verify_MAC(key=key, signature=signature, message=message)
    #print(f"Message is{'OK' if is_authentic else 'NOK'}")

    #2 zad
    PATH = "challenges/g3/jerkovic_ela/mac_challenge/"
    KEY = "jerkovic_ela".encode()

    messages = []
```

```
for ctr in range(1, 11):
    msg_filename = f"order_{ctr}.txt"
    sig_filename = f"order_{ctr}.sig"


    msg_file_path = Path(PATH + msg_filename)
    sig_file_path = Path(PATH + sig_filename)
    with open(msg_file_path, "rb") as file:
        message = file.read()

    with open(sig_file_path, "rb") as file:
        signature = file.read()

    is_authentic = verify_MAC(key=KEY, signature=signature, message=message)
    if is_authentic:
        messages.append(message.decode())

messages.sort(
    key=lambda m: datetime.datetime.fromisoformat(
        re.findall(r'\(.*?\)', m)[0][1:-1]
        )
)
for m in messages:
    print(f"Message is{m:>45}OK")
```